IDA PAPER P-3126

STRATEGIES AND IMPLEMENTATION ARCHITECTURES FOR SELECTED DEPARTMENT OF DEFENSE SOFTWARE REPOSITORIES

Audrey A. Hook, *Task Leader*

Michael C. Frame

DTIC ELECTE JAN 0 3 1996 S D F

19951228 041

September 1995

Prepared for

Office of the Deputy Assistant Secretary of Defense (Information Management)

Office of the Assistant Secretary of Defense
(Command, Control, Communications and Intelligence)

Approved for public release, unlimited distribution: December 5, 1995

DTIC QUALITY INSPECTED 1

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772

DEFINITIONS

IDA publishes the following documents to report the results of its work.

Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract DASW01 94 C 0054 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

© 1995 Institute for Defense Analyses

This material may be reproduced by or for the U.S. Government pursuant to the copyright license under the clause at DFARS 252.227-7013 (10/88).

IDA PAPER P-3126

STRATEGIES AND IMPLEMENTATION ARCHITECTURES FOR SELECTED DEPARTMENT OF DEFENSE SOFTWARE REPOSITORIES

Audrey A. Hook, *Task Leader*

Michael C. Frame

Accesion For
NTIS CRA&I
DTIC TAB
Unannounced
Justification
By
Distribution /
Availability Codes
Dist | Avail and / or Special
A-1

September 1995

Approved for public release, unlimited distribution: December 5, 1995

INSTITUTE FOR DEFENSE ANALYSES

Contract DASW01 94 C 0054
Task T-J5-1298

PREFACE

This paper was prepared by the Institute for Defense Analyses (IDA) under the task order, Information Technology Strategic Plan, and fulfills the objective of presenting alternative strategies for integration of DoD repositories. The work was sponsored by the Deputy Assistant Secretary of Defense (Information Management), Office of the Assistant Secretary of Defense (Command, Control, Communications and Intelligence).

A review of this paper was performed by the following IDA research staff members: Ms. Anne A. Douville, Dr. Edward A. Feustel, Dr. Dennis W. Fife, Dr. Richard J. Ivanetich, Dr. Richard P. Morton, and Mr. Glen R. White.

Table of Contents

List of Tables

EXECUTIVE SUMMARY

This paper reviews proposed integration strategies and implementation architectures for the integration of three Department of Defense software repositories: the Interim IDEF repository for process re-engineering; the Defense Data Dictionary System (DDDS), a repository for standard data elements, and the Defense Software Reuse System (DSRS), a repository for software components. A primary goal of this integration is to allow users a simple way of accessing the three repositories without being aware of which repository they are using or when they move from one repository to another.

Information needed to perform a detailed analysis was not available. Therefore, at the sponsor's direction, general assumptions were made by the Institute for Defense Analyses research team, and a relatively simple model was developed to evaluate each strategy-implementation architecture and how it met evaluation criteria.

Two strategies were given serious consideration: (1) the re-use integration strategy, and (2) the re-engineering integration strategy. The *re-use integration strategy* would provide an integrated view of component repositories without necessarily requiring major changes to those components. This would involve either a standards-based approach where the repository providers would standardize a common look-and-feel interface, or a mapping-based approach where the providers would develop a software layer that would provide a mapping to the underlying component repositories. The *re-engineering integration strategy* would involve designing and developing a completely new system, and having users make the transition from the three existing systems to the new system.

The four candidate implementation architectures capable of supporting the integration strategies would use one of the following technologies: (1) Hypertext Markup Language (HTML) and the World-Wide Web, (2) HTML and the Web in conjunction with a commercial database management system, (3) a commercial database management system alone, and (4) a software product specifically created to support repositories.

Technical, functional, administrative, and cost criteria were considered in judging the value of a particular strategy-implementation architecture. The evaluation approach and

model developed during the course of research scored and weighed three of the four criteria. The functional criteria were not evaluated within this model because their issues were the same for any strategy-implementation choice. The method and model were made generic enough to be applied to any other type of repository.

The evaluation model provided the basis for recommending the re-use integration strategy with the HTML-database implementation. This choice reflected our experiences with the Web as an effective vehicle for supporting distributed repositories. Although the HTML-database implementation scored only slightly higher than the next two combinations, it had the following distinct benefits:

- The three repositories (Interim IDEF, DDDS, and DSRS) are already on the Internet and will only have to set up a Web environment. This will provide a relatively fast implementation of integrated user interfaces. The Web supports the necessary distributed processing, and it is easy to construct Web home pages that provide hypertext links to the repositories. In addition, only a small number of programs are necessary to construct HTML versions of the repository holdings.
- The capability exists for future evolution to either of the other two high scoring alternatives. Since the view of the repositories will give the appearance of an integrated whole, any re-engineering integration of the underlying repositories will be invisible to the user. The effect that re-engineering integration will have on the HTML interface will be to change the links to the documents of the integrated repository. If these links were generated automatically (as they should have been), an underlying re-engineering integration will be accounted for by regenerating the higher-level pages.
- Support is now provided for a graceful evolution toward integration. The existing systems can continue to operate as they currently do and current users will not be affected. The new interface will be an add-on to the existing systems. New users will be able to use the new or the old interface. Existing users will be able to switch if they so desire.

Finally, a three-phase implementation approach is given for the evolution to the new user interface.

1. INTRODUCTION

In 1993, the Institute for Defense Analyses (IDA) performed a study for the Defense Information Systems Agency/Center for Information Management (DISA/CIM) to determine the feasibility of integrating the user interfaces of three software repositories [Cohen and Frame 1993]. Subsequently, DISA/CIM concluded that a single platform, centralized repository would yield integrated repositories that could be most effectively managed and accessed.

In 1994, the Office of the Deputy Assistant Secretary of Defense (Information Management) (DASD (IM)) asked IDA to explore a range of possible repository integration strategies, including the approach adopted by DISA/CIM. This paper presents alternative strategies and implementation architectures for integrating the three Department of Defense (DoD) repositories:

- Interim IDEF, a repository for process re-engineering.
- Defense Data Dictionary System (DDDS),[1] a repository for standard data elements.
- Defense Software Reuse System (DSRS), a repository for software components.

Current DISA/CIM efforts center around two of the three repositories, the DDDS and the Interim IDEF. These two repositories are the object of several integration projects at the DISA Center for Software Data Administration Department (CFSW/DAD), including the following:

- The acquisition of a commercial off-the-shelf (COTS) repository tool by the CFSW for building a growing repository. It will be used initially to manage an integrated DDDS and Interim IDEF. The software is expected to be acquired in the summer of 1995, and the integrated DDDS and Interim IDEF are scheduled to be operational in the fall of 1995.

---

[1] Formerly known as the DoD Data Repository System.

- The development of an integrated logical model of DDDS and Interim IDEF (called the Defense Integrated Repository System (DIRS) 2.0 model) by the CFSW. The schedule for completing this task is unknown at this time; however, the COTS repository tool requires it in order to be operational.
- The re-hosting of the Interim IDEF repository on the same machine as DDDS. Part of this effort is to merge Interim IDEF and the U. S. Army Corps of Engineers (USACE) Data Encyclopedia, currently called the CoE Encyclopedia. The Interim IDEF and the CoE Encyclopedia use the same logical and physical data models, and they share programs and utilities.

For this paper, the term *repository* means a source of information about things, and, possibly, a source of the things themselves. For example, a software repository may contain information about software as well as containing the software itself. *Integration* is defined as a merging of components in some way. In this paper, the goal of integration is to simplify user access to the three pertinent repositories, and to make it possible for a user to request information that involves crossing repository boundaries. Users should not have to be aware of which repository they are using, or when they are moving from one repository to another.

Information needed to perform a detailed analysis was not available or could not be obtained in a timely way. Therefore, as the sponsor requested, we made general assumptions about user requirements and developed a relatively simple model to evaluate alternatives. The model served as a basis for the final recommendation of a strategy-implementation architecture in a three-phase approach, and is sufficiently generic to be adapted to other types of repositories.

We reviewed documentation relating to the DDDS, Interim IDEF, and DSRS repositories; interviewed personnel at the CFSW/DAD; attended a Defense Data Repository (DDR) Steering Committee meeting; and reviewed minutes and slides from a previous DDR Steering Committee meeting. This information contributed to an understanding of current efforts related to implementing integrated repositories.

We also applied our knowledge of other repository efforts (e.g., the Model and Simulation Resource Repository of the Defense Modeling and Simulation Office) and of the technologies relating to repositories to develop a manageable number of alternatives. A set of criteria was developed to evaluate the alternatives, and a model was developed that involved assigning weights to the criteria and scores to each criterion for each alternative.

2. STRATEGIES, ARCHITECTURES, AND EVALUATION CRITERIA

The problem of analyzing strategies for repository integration is divided into three dimensions: the strategies themselves, alternative implementation architectures, and evaluation criteria. An implementation architecture may be applied to more than one strategy. Therefore, our approach was to compare each meaningful strategy-implementation architecture in terms of how well it meets the evaluation criteria. This section explains the three dimensions in detail.

2.1 Strategies

This report considers two major strategies: re-use integration and re-engineering integration. *Re-use integration* provides the user an integrated view of component repositories without necessarily requiring major changes to those components. *Re-engineering integration* restructures the underlying architecture.

A third strategy is to *do nothing*: continue to use and develop the existing repositories as separate entities. This has some short-term cost benefits and is the least disruptive to the organizations involved in supporting the existing repositories. However, assuming there is a need for integrated repositories, this strategy places the responsibility for retrieving and integrating information in the hands of users. This leads to a great deal of duplication of effort since each user would have to solve the integration problem individually and possibly repeatedly. CFSW/DAD has already begun pursuing integration efforts which make further discussion of this strategy pointless.

2.1.1 Re-Use Integration

Two general approaches to re-use integration are possible: standards based and mapping based.

Standards-based re-use integration requires the repository providers to standardize on a common look-and-feel interface. For example, since all three repositories of interest are implemented using Oracle, a user interface implemented with SQL*Forms could be

required, but that is not enough to achieve a uniform interface among repositories. Two repositories are already built upon this software, and yet the "look-and-feel" of the two systems is noticeably different (e.g., menu structure, menu selection mechanisms, key mappings). Unless the style of forms and menus and the flow from screen to screen are similar, a jolt is still experienced when the user goes from one repository to another. For example, when users move from one window to another, they have to remember a different key is used for making a menu selection in the system associated with that window.

Mapping-based re-use integration is achieved by developing a software layer that provides a mapping to the underlying component repositories. The software layer implements a unified logical data model that maps to the logical data models of the underlying repositories. Users have the impression of the unified logical model. However, this strategy also requires that the user interface be changed to use the single logical data model.

A very low-level kind of integration based on windowing was studied extensively by IDA in 1993 [Cohen and Frame 1993]. The repositories and their user interfaces could be left untouched, and the windowing capability of a personal computer or workstation could be used to open windows to the various repositories of interest. This approach has the advantage of being a "do nothing" alternative for the repository providers. The implementation could simply use existing windowing software and terminal emulator software running in windows. This can hardly be considered integration at all. In addition, it would either require users to obtain computers with particular windowing capabilities, or else would require extensive documentation and user support to allow for all the different windowing and terminal emulation systems that are available. Because windowing provides such a minimal gain, it was not considered further.

2.1.2 Re-Engineering Integration

The re-engineering integration strategy involves analyzing the requirements for all the existing component repositories, designing a new repository system that incorporates all these requirements, implementing the system, converting the information maintained in existing repositories, and having users make the transition from existing systems to the new system.

2.2 Architectures

The current state of hardware and software technology leads to a consideration of four major architectures capable of supporting the integration strategies through using the following technologies:

- Hypertext Markup Language (HTML) and the World-Wide Web (WWW, or simply "the Web") resources.
- The Web in conjunction with commercial database management systems (DBMSs).
- A commercial DBMS only.
- Software specifically created to support repositories.

The following sections give examples are given of how each architecture would be used.

2.2.1 Hypertext Markup Language

To appreciate the value of the Web for supporting repositories, it is helpful to understand the basic operation of the Web. The Web is a virtual client-server network on the Internet. An Internet node may be a Web client, a Web server, or both. It is easier to understand the Web from the client perspective initially.

Web clients request and receive "documents" from Web servers. These documents are usually in the form of HTML. HTML documents contain text and possibly graphics, and formatting commands to indicate how the information is to be displayed. In addition to controlling the appearance of a document on the screen, the HTML commands can specify that some parts of the document are links to other documents that can reside at any Web server. Browsers usually indicate the parts of the document that are links by underlining or some other type of highlighting. The user has the ability to "point-and-click" at the highlighted area. When the browser detects this action, it uses the information about the link to send a request for the associated document to the proper Web server. Once the Web server has transmitted the document back to the browser, the browser displays it to the user. The user can then read the new document and continue to follow new links. The browser also keeps track of where the user has been and supports returning to previously viewed documents.

This is a simplified view of the client operation. A client can receive different kinds of files and display them all, including text documents (as already described), forms to support data entry and display, large files to be downloaded to the client site, graphics, audio files, and other forms of multimedia data. Since the interface between the client and server is based on a well-defined protocol, it has been possible to build a number of browsers that support personal computers, more powerful workstations, and dumb terminals. The latter

obviously have a more limited display capability and would use a keyboard command instead of a mouse to select a link.

A Web server is a program that listens on an Internet port for messages. These messages are in the form of a request for a document and the address of the requester. The server locates the document and transmits it to the requester. It is possible for a request to be associated with a program to be executed. In this case, the server invokes the program and sends it the part of the request that corresponds to parameters to that program. The program executes and generates a document for the server to send back to the requester. This mechanism permits a database application to be invoked to select information from a database and create an HTML document that incorporates the dynamically selected database information.

This simplified overview of the Web is intended to explain how the Web supports distributed databases, device-independent user interfaces, and client-server processing.

One way of implementing a repository is to use the Web hypertext-oriented system. The repository information is maintained as a set of HTML documents. Tools are available (and can be enhanced or new ones developed) to process these textual documents, produce hierarchies of indexes to the documents, and to link documents together based upon common threads.

There are many "virtual repositories" in existence on the Web. These have been formed either intentionally, by organizing HTML documents to provide access to underlying information, or *ad hoc*, by finding related information on the Web and organizing documents that point to them. Each of these approaches is also supported by a number of existing information retrieval systems that permit rapid full-text searches (e.g., Wide Area Information System (WAIS), Lycos, Harvest, and other various "web crawlers").

2.2.2 HTML-Database Implementation

It is possible to maintain repository data in a database (or databases) and extract it into HTML documents. This approach is already in use at a number of locations on the Web. Many HTML-to-database interfaces have been developed for a number of commercial DBMS products.

This implementation architecture can be used in two possible ways (depending upon performance requirements):

- A "batch" process can be run periodically to generate both HTML documents with built-in links and index hierarchy documents from the database(s). The process of running the "batch" job can be optimized so that only modified information has to be regenerated. The frequency of running the batch process would depend upon the frequency of updates to the underlying database, the need for up-to-date information, and the cost of running the update.
- When a user selects a link, a query is executed to build the target HTML document in immediate response to the query.

2.2.3 Database Implementation

A database implementation architecture involves storing the structured data in database management systems and providing "structured" access to the data. This is how DDDS, Interim IDEF, and DSRS are currently implemented. In order to achieve integration, either the repositories are all stored in a single DBMS (centralized database), or the databases must be able to query one another and combine the results (distributed database). The distributed version of this architecture involves implementing a global data dictionary or providing the ability to share data dictionary information between systems. Many commercial DBMS products have implemented such a distributed database capability, and some have the ability to interface to other commercial DBMS products.

2.2.4 Repository Product

The use of software specifically designed to support repositories is an approach being pursued by the DISA CFSW to achieve an integration of DDDS and Interim IDEF. It involves acquiring a COTS repository product (currently, two candidates are Rochade and InfoSpan).

Repository products that support software engineering maintain metadata about the information being managed. They usually provide a modeling capability so that the metadata can be used to establish relationships. The repository product also provides a storage capability for the metadata. The storage is usually implemented using a commercial DBMS product. In addition to providing support for management of an enterprise and secure access to repository components, a repository product provides a framework to allow interaction among other products [Sharon and Bell 1995].

The major advantage of a repository tool is that it is specifically built to support repositories. However, repository tools are not well defined (there is no standard set of features that defines a repository tool), so it is difficult to make general statements about them.

Some repository tools are "software backplanes" that support the integration of software tools. In this case it is possible that existing databases could be "plugged in" to the backplane.

2.3 Evaluation Criteria

A strategy may be realized by any of a number of implementation architectures. Judgements about the value of a particular strategy and implementation architecture can be made by rating them based upon how well they address technical, functional, administrative, and cost criteria. Those criteria are explained in the remainder of this section.

2.3.1 Technical Criteria

Any technical approach to integration should be evaluated in terms of how complete an integration can be achieved, how scalable the approach is to the growth of components added to existing repositories or the addition of new repositories, how flexible the approach is in handling new types of repository information, and how well the approach allows for the adoption of useful new technology as it becomes available.

Level of integration refers to the four possible levels of integration (from lowest to highest):

- No integration.
- The ability to easily move from one repository to another.
- The ability to use results from one repository to search other repositories.
- Full integration (the user is not aware of repository boundaries).

Scalability refers to the extent to which an alternative supports growth in the number of component repositories, in the demand on those repositories, and in the size of those repositories.

Flexibility refers to the extent to which an alternative supports future changes in the requirements placed on integrated repositories and how well it permits adoption of new technologies as they become available.

2.3.2 Functional Criteria

Requirements for repositories to be integrated with other repositories must be determined because these requirements will affect both the re-use integration and re-engineering integration strategies. However, requirements have more effect on the re-engineering inte-

gration strategy since it requires *new* analysis and design. Understanding the functional requirements for each repository and for integrating repositories is essential for analysis and design. Appendix A presents questions that should be addressed when developing functional requirements.

Because the focus of this task is repository integration, there are certain generic things that can be said about functional criteria:

- Users should not have to be aware of repository boundaries. The set of repositories should appear as a unified whole to the user.
- Proponents for repositories should not lose any administrative control when repositories are integrated.

2.3.3 Administrative Criteria

Administration may be centralized or distributed. Administration includes determining who is allowed access to data, what kinds of access are permitted, setting and enforcing the policy for how updates are applied, and controlling the availability of the data.

Another administrative concern is security. Security problems and solutions are generally the same for all strategies and implementations. Therefore, security will not be considered as a differentiating criterion here.

2.3.4 Cost Criteria

Costs are incurred for operating and maintaining a system and for performing integration. In addition, users incur costs in using a system. Therefore, cost criteria may be categorized as follows:

- *Cost of system operation and maintenance.* This is the cost of operating the component repositories as well as the integration architecture, and of maintaining the integrated system as problems are detected or requirements change.
- *Cost of integration effort.* This is the cost of integrating an existing system or developing a new system. It includes the cost of making changes to existing systems and acquiring any additional technology needed to complete the integration.
- *Cost incurred by users of the system(s).* This is the cost of using a repository system. There may be a one-time cost as well as a per-use cost.

3. EVALUATION MODEL

The evaluation model scores and weighs three of the four major sets of criteria: technical, administrative, and cost. The functional criteria are not evaluated within this model because their issues are the same for any strategy-implementation architecture choice.

Section 3.1 presents a discussion of the evaluation approach developed for the study. Section 3.2 describes the model that was developed using this technique. Section 3.3 contains the scoring rationale behind the weighting and scoring. Section 3.4 identifies the advantages and disadvantages of each strategy-implementation architecture.

3.1 Evaluation Approach

Weighting. Each criterion is given a weight to indicate its relative importance in the evaluation. All the weights should sum to 100.

If a criterion is further subdivided, its subdivisions should be assigned weights which sum to the weight assigned to the criterion. For example, if criterion X is assigned a weight of 20 and is subdivided into X1 and X2, then X1 might have a weight of 5 and X2 a weight of 15. These weights indicate the relative importance of a criterion. In the example, X2 is three times as important as X1. The use of weights permits criteria that are not quantitatively comparable to be considered together.

Scoring. Once the weights have been assigned, it is necessary to score each criterion for each strategy-implementation alternative. Whenever possible, scores should be based upon quantitative information. For example, if the budget for an item is known, then the score for the cost of providing that item should be based on how well the cost adheres to the budget.

In the model presented in this paper, a "good" score should always be higher than a "bad" score. Since a low cost is generally considered better than a high cost, a low cost must be converted to a high score. If one alternative has a cost that is 80% of the budgeted cost, and another has a cost that is 50% of the budgeted cost, the inverse of this percentage might

be used to compute a score. In this case the lower cost would get a score of 2 and the higher cost a score of 1.25. This score should be further normalized so that an extremely high (or low) score does not throw off the overall score. In this example, normalization might involve using each "raw" score divided by the maximum "raw" score; so, a raw score of 2 becomes a score of 1 and a raw score of 1.25 becomes a score of 0.625.

If it is not possible to obtain quantitative information in arriving at a score, then a more subjective approach can be used. For example, create a number of score ranges and assign a score based upon which range the criterion is believed to fall into. A simple example is to use "high" (score = 3), "average" (score = 2), and "low" (score = 1). This kind of scoring may be used for such criteria as the financial viability of a supplier.

3.2 Evaluation Model

Weighting. For this study, each criterion is given a weight to indicate its relative importance in the evaluation: technical (40), administrative (10), and cost (50). The technical and cost criteria are further refined: technical is made up of six sub-criteria, cost is made up of three. Their weights are distributed across these sub-criteria. For example, the sum of the weights of the six technical sub-criteria is 40.

Scoring. The technical and administrative criteria are each scored as either 3 (high), 2 (average), or 1 (low). As the administrative criterion is not further subdivided, a high score is assigned if the potential easily exists for decentralized control; a low score is given if decentralized control would be difficult or impossible to achieve. Cost and its sub-criteria are also scored as high, medium, and low, but their numeric values are reversed: a high cost gets a lower score than a low cost.

Evaluation matrix. The weights, scoring, and results are summarized in Table 1, Evaluation Matrix, on page 13. The matrix has a column for each strategy-implementation architecture. The re-use integration strategy assumes an attempt to provide an interface to the existing databases. Therefore, the HTML architecture is not considered since it would involve major changes to the databases.

Operations and Maintenance (O&M) cost is assumed to be roughly the same for each strategy and implementation.[2] COTS versions of Web software, for example, browsers and servers from Netscape, have become available and their providers are offering technical support to the buyer.

---

[2] The existing repositories were budgeting between $160,000 and $200,000 each in FY93 for O&M.

Table 1. Evaluation Matrix

Concern	Weight	Re-Use Integration		Re-Engineering Integration		
		HTML-DB	DB	HTML	HTML-DB	DB
Technical (40)						
Level of integration						
Front-end integration	10	3	3	3	3	3
Logical DB	8	1	1	3	3	3
Physical DB	4	1	1	3	3	3
Scalability	6	3	2	3	3	3
Flexibility	12	3	2	3	2	2
Administrative (10)	10	3	2	3	3	3
Cost (50)						
O&M	15	2	2	2	2	2
Integration	15	3	2	1	1	1
User	20	3	3	3	3	2
Score (300 possible)		261	218	255	249	223
% Score		87	73	85	83	74

The choice of criteria, the assignment of weights to the criteria, and the assignment of scores to the criteria for each strategy-implementation architecture are the result of our best judgement. Given the uncertainty of some of the factors, we wish to present an evaluation where the decision process is clear and easy to understand, even though some will not agree with the judgements, or the result. The use of three scores (3, 2, 1) to represent general "goodness," and the use of weights to assign priorities to the criteria, yields a simple representation that captures everything that went into the evaluation. *Why* certain weights and scores were assigned is discussed in the following paragraphs.

Weight Assignment. The technical quality of an implementation and the cost of operating and converting to an integrated system were considered to be paramount. In these days of elevated cost consciousness, a slightly higher weight was assigned to cost. Since the whole point of integration is to make the system more useful to users, the cost to the user gets a slightly higher weight than each of the other costs. By the same token, technical

criteria that affect integration get a higher score than the others (e.g., scalability.) Administrative concerns count for 10% of the overall score for each alternative. Although it is possible that current providers will want to give up some of their autonomy (with respect to their repositories), it seems more conservative to give a better score when autonomy (decentralized administration) is preserved. In addition, this agrees with one of the generic functional criteria mentioned in Section 2.3.2 on page 8.

Score Assignment. The reason for the different scores should be clear from the explanation of the implementations and the criteria. For a particular criterion, the same score was assigned to two implementations if the effort, value, or cost was felt to be roughly comparable. Different scores were assigned when it seemed there would be a significant difference.

Note: The use of only three scoring levels means that a significant distinction must exists before two alternatives would get different scores. A more detailed evaluation based on better quantitative data should use ten or even one hundred scoring levels so that finer distinctions can be captured.

3.3 Scoring Rationale

Re-engineering integration strategy. This strategy has all the costs associated with software development. Some of this work is going on now, such as design of a consolidated database. It is still necessary to acquire the software that will support the new repository, develop the integrated system, produce documentation, conduct training, convert the existing repositories to the new one, have users make the transition to the new system, and conduct a period of parallel operation. In addition, the single integrated repository implies a change in the administrative control of the systems.

The re-engineering integration strategy has the advantage of providing an opportunity to rethink previous decisions and to learn from previous mistakes. It may also provide a much less expensive operational system since such items as software licenses, personnel, and facilities might no longer be duplicated. It would be possible to provide a single user interface and to provide a single point of support for users. On the other hand, user access is delayed and cost is high.

The re-engineering integration strategy does not fit well with the current ideas and practices of rapid prototyping and evolutionary development. This is accounted for by assigning generally lower scores for integration cost.

The re-use integration strategy and the HTML-DB. This implementation architecture allows existing repositories to continue to operate exactly as they do now. Each system will have to acquire, install, and manage Web server software. Web server software may be obtained for free; however, there is no consistent form of support for the public domain version. There are many informal support resources available over the Internet, such as the `comp.info.systems` newsgroup. Web servers are now available as COTS products from vendors who provide full support for license holders. The major cost will be the personnel cost of training and supporting Web administrators. The systems currently under consideration already have Internet connections.

The Web implementation architecture provides for faster user access to integrated repositories, at lower cost, and with a phased, low-risk path to full integration, as is reflected in higher scores for user cost, flexibility, and technology use.

3.4 Implementation Architectures

Hypertext Markup Language (HTML). The use of a Web browser has the advantage of providing a multi-level solution to the re-use integration strategy. On the one hand, the same kind of browser software (e.g., Netscape, Mosaic) is used for all applications, and important items within a display look the same. On the other hand, the content of a document might look different, depending on which repository application generated it. While this content difference might not be desirable in the long run, the mapping-based approach provides integration sooner and supports evolution toward a more uniform appearance of the repository application-generated documents. In addition, it provides immediate support to users with many kinds of hardware, such as dumb terminals, personal computers, and workstations. It also provides immediate integration with other sources of data and supports a multi-media interface for terminals that have that capability.

The Web provides a common look-and-feel interface within the boundaries of HTML documents. The content of a message will be application dependent, but even here there is some consistency in the way links and fill-in-the-blank entries are displayed. Users who start working with HTML documents quickly get used to the variations and are able to discern the regularities. Users will be able to use their favorite browsing tools, not just the one provided by the repository. In addition, they will be able to search other resources on the Web while they are using the repositories. Thus, information providers can add additional documents to their systems which complement but are not part of their repository, without affecting the normal operation of the repository. For example, DDDS could provide

an HTML version of Department of Defense Manual DoD 8320.1-M. (DISA CFSW is already providing this document in machine-readable form on CD-ROMs as part of its PCAT (PC Access Tool) distribution.) IDEF could provide postscript versions of model graphics to be downloaded or viewed by repository users. Tutorials and documentation can all be maintained on line.

The preceding comments apply to any architecture that is based upon the Web. In addition, there are advantages and disadvantages that are unique to an HTML implementation.

Advantages

- It leverages off the large community of Web developers.
- Scalability and flexibility are built in.
- Both centralized and distributed administration are possible.
- It integrates smoothly with other government agency repositories that are using the Web.

Disadvantages

- Network performance may be a problem if a Web site becomes very popular or has a slow link to the rest of the Web. However, this is a problem for any architecture that uses the Internet or runs on a heavily loaded or under-powered platform.

HTML-Database Implementation. All the advantages relating to using documents in HTML, mentioned in the previous section, are also valid here. In addition:

Advantages

- It is possible to continue to use existing repository databases and add a Web user interface.
- The repository application providers do not have to abandon their proprietary interfaces for users who wish to continue to use them.
- Use of a DBMS to support structured data provides better control over data quality and consistent updates.
- A DBMS provides good facilities for building links and document indexes.
- HTML style conventions can be built into the DBMS extraction tools.

- Centralized and decentralized administration is still possible.
- It is possible to combine this with the HTML approach so that some providers could maintain their data in databases while others maintain it in the form of HTML documents.

Disadvantages

- Proprietary DBMSs, if they are not already being used, are required unless new repositories are permitted to use an HTML-only implementation.
- Software for building HTML documents from database queries must be developed (some Web users have already developed such software that can be used as a starting point).[3]

Database Implementation Architecture. A database implementation has the database advantages and disadvantages of the HTML-Database implementation. However, a major additional disadvantage is that an integrated user interface must still be established. This requires that user interfaces provided by the DBMS vendor(s) be integrated in some way or that a third-party user interface product be used. Such products as Powerbuilder have the capability of interfacing to homogeneous as well as heterogeneous databases.

[3] Oracle Corporation has announced a Web Interface Kit (February 7, 1995). This seems to be largely a collection of the tools provided by others over the Web. Such tools are actually available for other DBMS products as well, but do not have such a tacit endorsement from the product vendors.

4. RECOMMENDED STRATEGY-IMPLEMENTATION ARCHITECTURE AND PLAN

Table 1 on page 13 shows that three strategy and implementation architecture combinations have roughly comparable scores:

- Re-use integration strategy with HTML-DB (87%)
- Re-engineering integration strategy with HTML (85%)
- Re-engineering integration strategy with HTML-DB (83%)

Although the first combination has a score that is only slightly higher than the other two, it is the preferred choice:

- The three repositories (Interim IDEF, DDDS, and DSRS) are already on the Internet, and will only have to set up a Web environment. This will provide a relatively fast implementation of integrated user interfaces. Web supports the necessary distributed processing, and it is easy to construct Web home pages that provide hypertext links to the repositories. In addition, only a small number of programs are necessary to construct HTML versions of the repository holdings.
- It allows for future evolution to either of the other two high scoring alternatives. Since the view of the repositories will give the appearance of an integrated whole, any re-engineering integration of the underlying repositories will be invisible to the user. The effect that re-engineering integration will have on the HTML interface will be to change the links to the documents of the integrated repository. If these links were generated automatically (as they should have been), an underlying re-engineering integration will be accounted for by regenerating the higher-level pages.
- It supports a graceful evolution toward integration. The existing systems can continue to operate as they currently do and current users will not be affected. The new interface will be an add-on to the existing systems. New users will be

able to use the new or the old interface. Existing users will be able to switch if they so desire.

Implementation Phases. The new user interface would be implemented in three phases.

Phase I - Web Software Implementation

- Each existing repository computer will become a Web server.
- A set of HTML documents will be developed that serve as a menu and data retrieval interface. The highest-level document (the home page) will initially allow users to branch off to one of the participating repositories. Each repository will have its own set of predefined HTML documents to support interaction with its system.
- Each system will adapt an Oracle-to-HTML interface (such software is available) to support querying the Oracle database and turning the query result into an HTML document that is displayed to the user. (The use of Oracle retains the investment in the current repository implementation.)

The Phase I integrated repository would not provide very good integration in the sense of being able to jump from one repository to another based on information gleaned from a repository. To do that would require a logical consolidation of the repositories as specified in Phase II.

Phase II - Logical Integration

- A consolidated data model will be developed that incorporates DDDS, DSRS, and Interim IDEF. This model (already under development at DISA CFSW for DDDS and Interim IDEF as the DIRS 2.0 model) will support integration across the databases. For example, an attribute in DDDS relating to an attribute in Interim IDEF will be defined over the same domain as the Interim IDEF attribute. Thus, a value retrieved in DDDS can be used to do a search in IDEF (and vice versa). In addition, duplication and redundancy in the databases can either be eliminated or capitalized upon.
- The HTML documents and the Oracle-to-HTML interfaces should be adapted to use the consolidated data model. Existing HTML documents can continue to be used. SQL views can be used to implement the consolidated data model until the underlying databases are actually modified.

The Phase II version will allow the repositories to be logically linked. Thus, it will be possible for a query to be processed by one database which will build an HTML document containing links to the other repository databases. By selecting a link, the user will be generating a query to a different underlying repository.

Phase III Physical Integration

Succeeding phases could involve re-engineering the database component of the integrated repository. The work of these later phases can be determined after the integration has begun and user feedback has been gained. It is possible a decision will have to be made to leave the databases alone, to physically move them into a single DBMS, to move them to other sites, or to replace the DBMSs with a repository tool. All these "back end" decisions can be made later, based upon cost and performance considerations, without affecting the user communities' functional access to the virtual integrated repository.

The administrative control of the participating repositories can remain localized with the current repository providers. A further advantage of the Web approach is that existing Web resources can be used to help make the individual and integrated repositories better. For instance, users can use electronic mail during a session to generate a message to repository administrators, indicating problems, asking question, or making suggestions.

APPENDIX A.
FUNCTIONAL REQUIREMENTS ANALYSIS

This appendix presents an approach for collecting and evaluating the functional requirements for the repositories, and performing a more detailed cost-benefit analysis. This approach is organized under the following steps.

Step 1. Identify candidate repositories and their providers (advocates).

Step 2. Analyze the functional requirements for the candidate repositories to determine potential groupings (repositories that might provide benefits by being combined in some way).

Step 3. Identify a set of "next generation" repositories. These are repositories that are expected to exist as a result of the integration effort. They may include existing individual repositories carried over "as is," and groups of existing repositories brought under an integrated repository umbrella by one of the integration strategy and implementation architecture combinations. Some repositories might be recommended for elimination as a result of this step.

Step 4. Collect economic information from the existing repository providers. This includes sufficient "raw" data to be used in an economic model of implementation architectures as well as summary data about the cost and benefit of the existing repositories. The cost and benefit information should be projected for *n* years and the basis for that projection should be explained. Costs should include operations and maintenance costs (facilities, personnel, software, hardware), expected use of the system (types of users, frequency of use, level of use), planned upgrades to the system, and cost and benefit to users.

Step 5. Perform a similar analysis of the implementation architectures. This analysis must include the cost of converting the existing systems to the candidate architectures.

Steps 1 through 3 may be partially addressed by conducting a groupware meeting, as suggested by DASD (IM). The following questions are intended as the basis for an agenda for such a meeting to discuss the need and capability for integrating repositories.

- Can we agree on what we mean by "repository"? DISA CFSW uses as a working definition "a specialized application that provides for shared storage and common access for data and objects required to support enterprise information systems and database development and reengineering" [Palmer 1995].
- Are other definitions needed? Can we agree on what we mean by "repository integration"?
- Do we want to achieve distribution of information about repository holdings?
- Do we want to achieve distribution of repository holdings?
- What is the list of repositories for consideration? (Candidate repositories are listed in Table A-1 on page A-4.)
- Do information processing standards for repositories exist?
- What are the requirements for each of the repositories?
- Who are the expected users?
- What is a scenario for repository use?
- What do users get from accessing the repositories?
- How are the repositories logically related? (A group at CFSW is developing an integrated logical model of DDDS and Interim IDEF. Is this an "as is" model, or a "to be" model? Is this modeling effort intended to incorporate other repositories later?)
- What are the requirements of these combinations for integrating combinations of repositories?
- Who are the expected users of these combinations?
- What is a scenario for combined repository use?
- What repository efforts are on going? (For example, design of new repositories, analysis of existing repositories, design of replacements for existing repositories, planning for integration of existing repositories, and implementation architectures of existing repositories. Aside from this IDA study of repository

integration, there are CISA CFSW's integration of DDDS and Interim IDEF; the DDR steering committee; the DISA CFSW integrated logical model, DIRS 2.0; and DISA CFSW's acquisition of an repository system compliant with the Information Resource Dictionary System (IRDS) standard.)

- Who is doing the work?
- Are the various projects complementary?
- Assuming repository integration needs are defined, who should do the work?
- How should it be accomplished?
- Should repository software be used?
- Should database software be used?
- Should Web software be used?
- Should custom-written software be used? If so, what modules?
- Should repositories be physically integrated?
- Should repositories be modified for integration?

Table A-1. DoD Repositories[a]

Name	Proponent	Remarks
Automated Resources Management System (ARMS)	Defense Automation Resources Information Center (DARIC)	Operational
Corporate Database	Defense Information Technology Services Organization (DITSO)	Planned Static information about operational units that are transferred to it (DITSO)
Computer-Assisted Software Engineering (CASE) Database	No proponent identified	Planned Will contain survey data about CASE tool use in the DoD IM community
Defense Data Repository System (DDDS)	DISA Center for Software (DISA/CFSW)	Operational Metadata about DoD data elements (legacy, candidate, and standard)
DoD IDEF Repository	Office of the Director, Defense Information Business Process Improvement Program	Operational Contains many DoD process and data models
Defense Software Repository System (DSRS)	DISA Center for Information Management (DISA/CIM)	Operational Contains information about software components

a. Source: ITAM Policy Planning Session, July 15, 1994.

LIST OF REFERENCES

Cohen, Brian S. and Michael C. Frame. DISA/CIM Repository Integration Evaluations. White paper. Alexandria, VA: Institute for Defense Analyses, 1993.

Information Technology Asset Management (ITAM) Policy Planning Session. ANDRULIS Research Corporation, July 15, 1994.

Palmer, Carl. Current Status of Defense Data Repository: Briefing to the DDR Steering Committee. May 16, 1995.

Sharon, David and Rodney Bell. Tools That Bind: Creating Integrated Environments. *IEEE Software* Vol. 12, No. 2 (March 1995): pp. 76-85.

LIST OF ACRONYMS

ARMS	Automated Resources Management System
CASE	Computer-Assisted Software Engineering
CD-ROM	Compact Disk - Read-Only Memory
CFSW	Center for Software
CIM	Center for Information Management
CoE	Corps of Engineers
COTS	commercial off-the-shelf
DAD	Data Administration Department
DARIC	Defense Automation Resources Information Center
DASD (IM)	Deputy Secretary of Defense (Information Management)
DB	database
DBMS	database management system
DDDS	Defense Data Dictionary System
DDR	Defense Data Repository
DIRS	Defense Integrated Repository System
DISA	Defense Information Systems Agency
DITSO	Defense Information Technology Services Organization
DoD	Department of Defense
DSRS	Defense Software Repository System
FY	fiscal year
HTML	Hypertext Markup Language
IDA	Institute for Defense Analyses
IDEF	Integrated Computer-Aided Manufacturing (ICAM) Method
IRDS	Information Resource Dictionary System

ITAM	Information Technology Asset Management
O&M	Operations and Maintenance
PCAT	PC Access Tool
USACE	US Army Corp of Engineers
WAIS	Wide Area Information System
WWW	World-Wide Web

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Final

4. TITLE AND SUBTITLE
Strategies and Implementation Architectures for Selected Department of Defense Software Repositories

5. FUNDING NUMBERS
DASW01-94-C-0054

Task Order T-J5-1298

6. AUTHOR(S)
Audrey A. Hook, Michael C. Frame

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

8. PERFORMING ORGANIZATION REPORT NUMBER
IDA Paper P-3126

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
DASD(IM)
Suite 910
1225 Jeff Davis Hwy.
Arlington, VA 22202

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for final release, unlimited distribution: December 5, 1995.

12b. DISTRIBUTION CODE
2A

13. ABSTRACT (Maximum 200 words)

This document reviews proposed alternative strategies and implementation architectures for the integration of three DoD software repositories: the Interim IDEF repository for process re-engineering, the Defense Data Dictionary System (DDDS) repository for standard data elements, and the Defense Software Reuse system (DSRS) repository for software components. Due to the lack of obtainable information about user requirements, the sponsor asked IDA to make general assumptions; consequently, an evaluation method and model were developed to evaluate alternative combinations of strategy and implementation architectures. The model served as the basis for recommending the re-use integration strategy with a Hypertext Markup Language database implementation architecture. The benefits of this implementation include providing a relatively fast implementation of the integrated repositories and support for a graceful evolution toward actual integration. A plan for a three-phase implementation approach is also given for the evolution to the new user interface and implementation architectures.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Interim IDEF, DDDS, DSRS, Software Repository, HTML, World-Wide Web, Interfaces, Integration, Apparent Integration Strategy.	46
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR